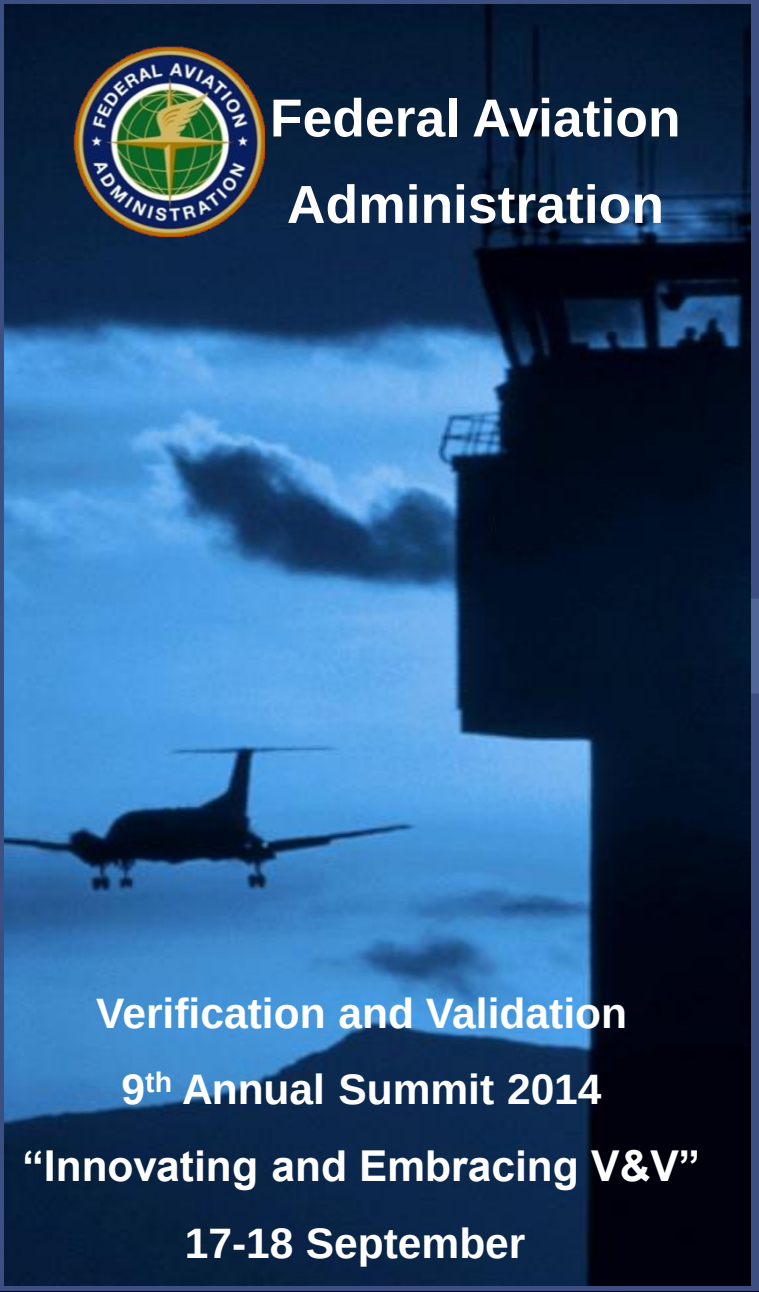



# Struggles at the Frontiers of Software Verification and Validation (V&V) for Software Intensive Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Kenneth E. Nidiffer



© 2014 Carnegie Mellon University

# SEI Background

Funded by the U.S. government as a research & development lab; (FFRDC)

Created in 1984 and administered by Carnegie Mellon University

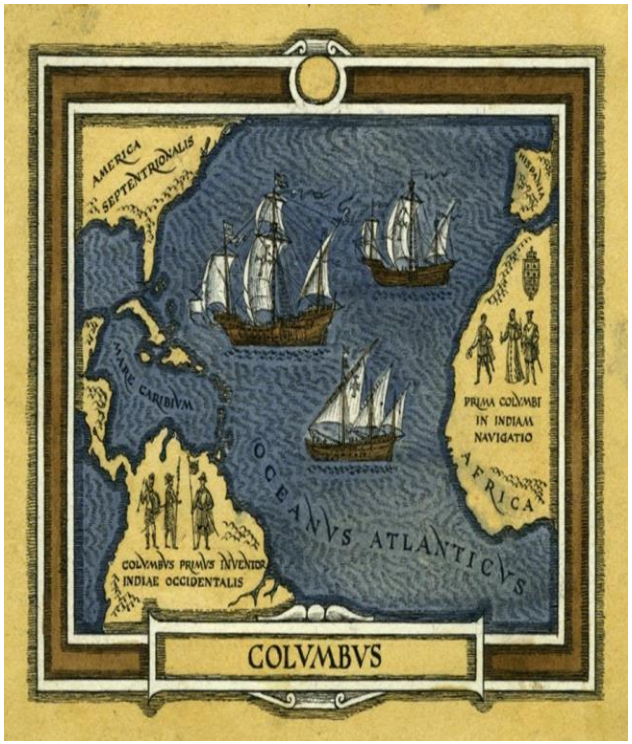
Headquartered in Pittsburgh, Pennsylvania; offices and support worldwide



© 2014 Carnegie Mellon University

# Topics

- Frontiers and Struggles
- What is Software V&V?
- The Challenges of Software V&V at the Enterprise Level
- Why is Software V&V Becoming More Important?
- Summary - Five Software V&V Keys for Success




**Source: SEI**


© 2014 Carnegie Mellon University

Federal Aviation Administration

# Why We Struggle with Software Verification and Validation (V&V)?



© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)?

According to Fred Brooks* software projects are difficult because of accidental and essential difficulties

- Accidental difficulties are caused by the current state of our understanding
  - of methods, tools, and techniques
  - of the underlying technology base
- Essential difficulties are caused by the inherent nature of software
  - invisibility - lack of physical properties
  - Complexity – for its size
  - conformity
  - changeability

*** *The Mythical Man-Month* by Fred Brooks, Addison Wesley, 1995**


© 2014 Carnegie Mellon University

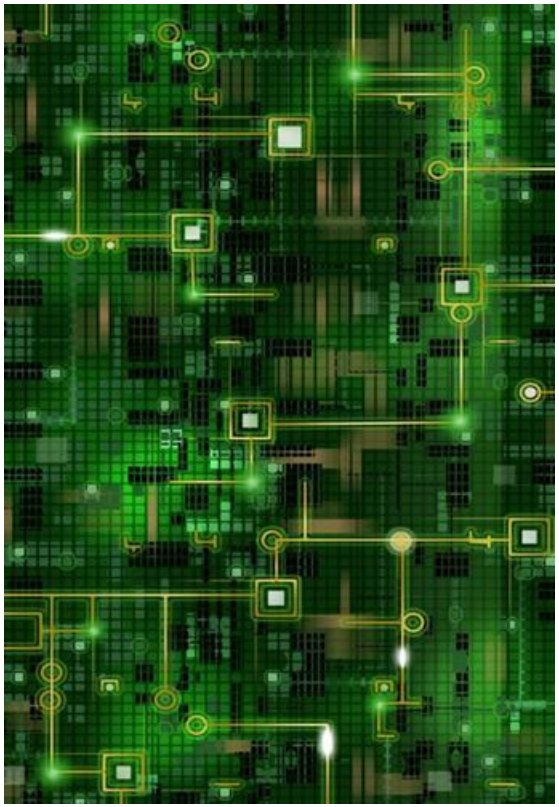
# Example: Why We Struggle with Software Verification and Validation (V&V)?

## Complexity:

Due to interaction of components, number of possible states grows much faster than lines of code.

For its size, software is very complex compared to other engineering artifacts

Hardware is complex but we usually know that for a known input, what to expect output should be



**Source SEI**



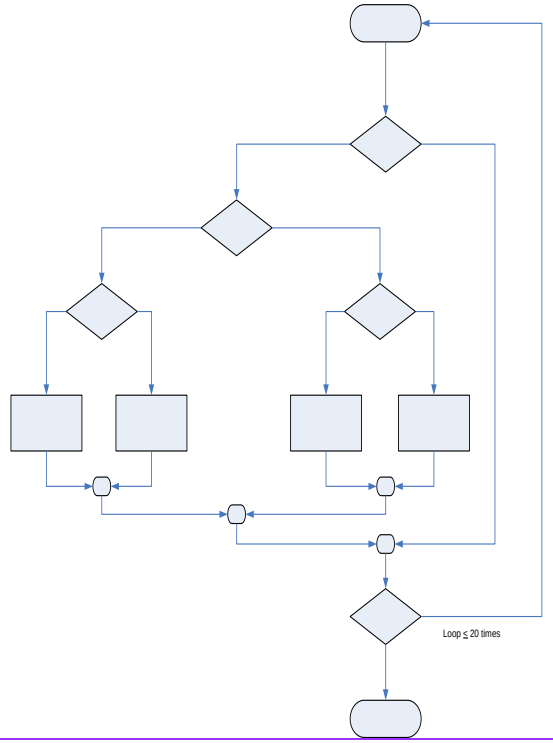
© 2014 Carnegie Mellon University

# Example: Why We Struggle with Software Verification and Validation (V&V)

**The flowchart might correspond to a 100 LOC module with a single loop that may be executed no more than 20 times.**

**There are approximately $10^{14}$ possible paths that may be executed!**

**For any but the smallest programs, complete path coverage for defect detection is impractical.**




**Lehman Laws:**

**1. The Law of Continuing Change – programs must change to be useful**

**2. The Law of Increasing Complexity – programs that change become more complex**

Adapted from Pressman, R.S., *Software Engineering: A Practitioner's Approach, Third Edition*, McGraw Hill, 1992



© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## We Are What We Were When

Source: SEI



© 2014 Carnegie Mellon University

# Example: Why We Struggle with Software Verification and Validation (V&V)

| | PHYSICAL SCIENCE | BIOSCIENCE | COMPUTER/SOFTWARE/CYBER SCIENCE |
|---|---|---|---|
| **Origins/History** | Begun in antiquity | Begun in antiquity | Mid-20th Century |
| **Enduring Laws** | Laws are foundational to furthering exploration in the science | Laws are foundational to furthering exploration in the science | Only mathematical laws have proven foundational to computation |
| **Framework of Scientific Study** | Four main areas: astronomy, physics, chemistry, and earth sciences | Science of dealing with health maintenance and disease prevention/treatment | • Several areas of study: computer science, software/ systems engineering, IT, HCI, social dynamics, AI<br>• All nodes attached to/relying on netted system |
| **R&D and Launch Cycle** | 10-20 years | 10-20 years | Significantly **compressed**; solution time to market needs to happen very quickly |

**Source: SEI**

**HCI: Human Computer Interaction; AI: Artificial intelligence**


© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Now We Look at Frontiers Differently

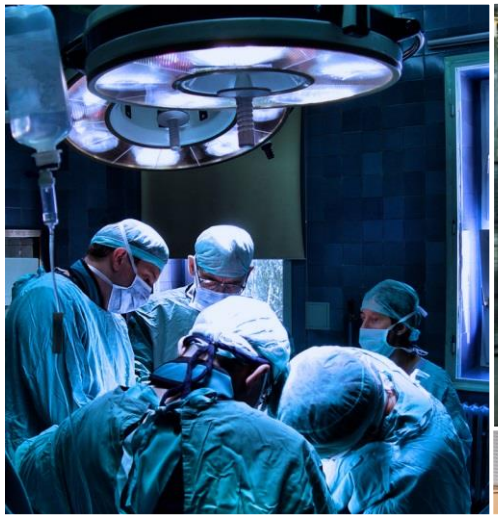
Source: SEI

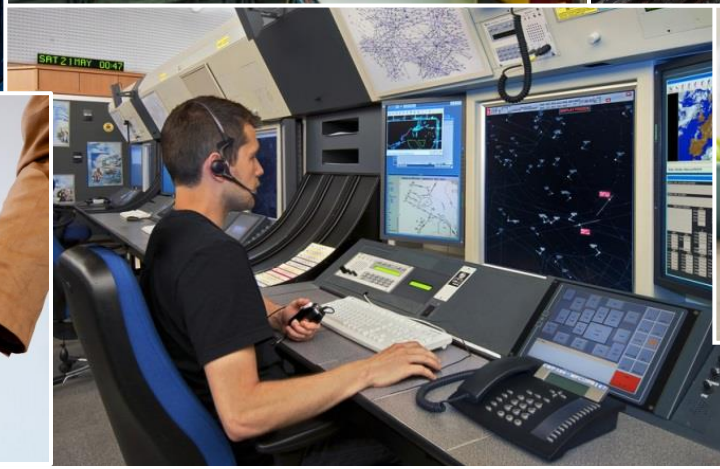


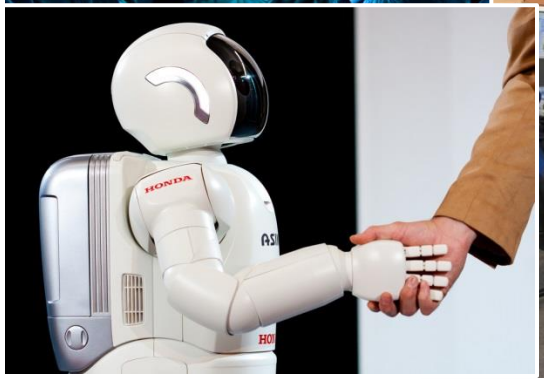
© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Software is Everywhere







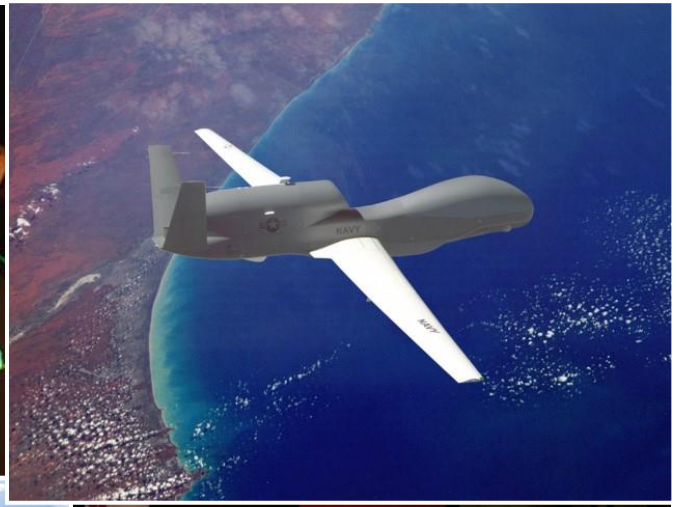
**Source: SEI**

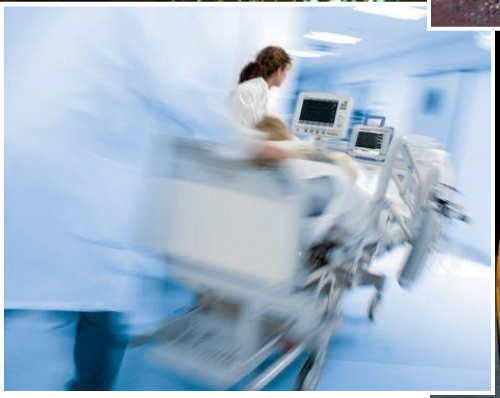


© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Software is Increasingly Complex




Source: SEI



© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Software Connects Us


Source: SEI



© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Software is Becoming More Personal




**Source: SEI**



© 2014 Carnegie Mellon University

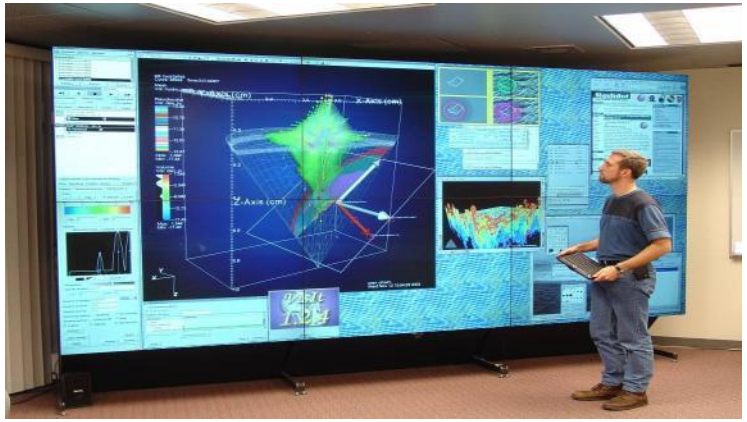
# Why We Struggle with Software Verification and Validation (V&V)

## Software is Important

Manufacturing

Finance

Space and Aviation



Engineering

Source: SEI



© 2014 Carnegie Mellon University

# Why We Struggle with Software Verification and Validation (V&V)

## Software is Today's Strategic Resource

Source: SEI



© 2014 Carnegie Mellon University

Federal Aviation Administration

# What is software V&V?



© 2014 Carnegie Mellon University

# Verification and Validation (V&V)*

Verification ***(Are we building the product right?)***

- The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase
- Is internally complete, consistent and correct enough to support next phase
- The system meets the validated specification!

Validation ***(Are we building the right product?)***

- The process of evaluating software throughout its development process to ensure compliance with software requirements. This process ensures:
  - Expected behavior when subjected to anticipated events
  - No unexpected behavior when subjected to unanticipated events
  - System performs to the customer's expectations under all operational conditions
- The system meets the operational need!

* Reference: IEEE Standard for Software Verification and Validation (1012-2004)


© 2014 Carnegie Mellon University

Federal Aviation Administration

# The Challenges of Software V&V at the Enterprise Level



© 2014 Carnegie Mellon University

# FAA OV-1





© 2014 Carnegie Mellon University

# The Challenges of Software At the Enterprise Level

## Today's Development Challenges

**Huge system/software engineering endeavors in aircraft, space vehicles, command and control, ground infrastructure, battle management, etc.**

- Several million SLOC programs; "hybrid" systems combining legacy re-use, COTS, new development
- Multi-contractor teams using different processes; dispersed engineering, development & operational locations
- New technologies create opportunities/challenges; products change/evolve, corporations mutate
- Business/operational needs change - often faster than full system capability can be implemented
- Skillset shortfalls; cost and schedule constraints
- Demands for increased integration, interoperability, system-of-systems capabilities
- Enterprise perspectives/requirements; sustainment concerns
- Software increasingly connects other systems

**Source: SEI**

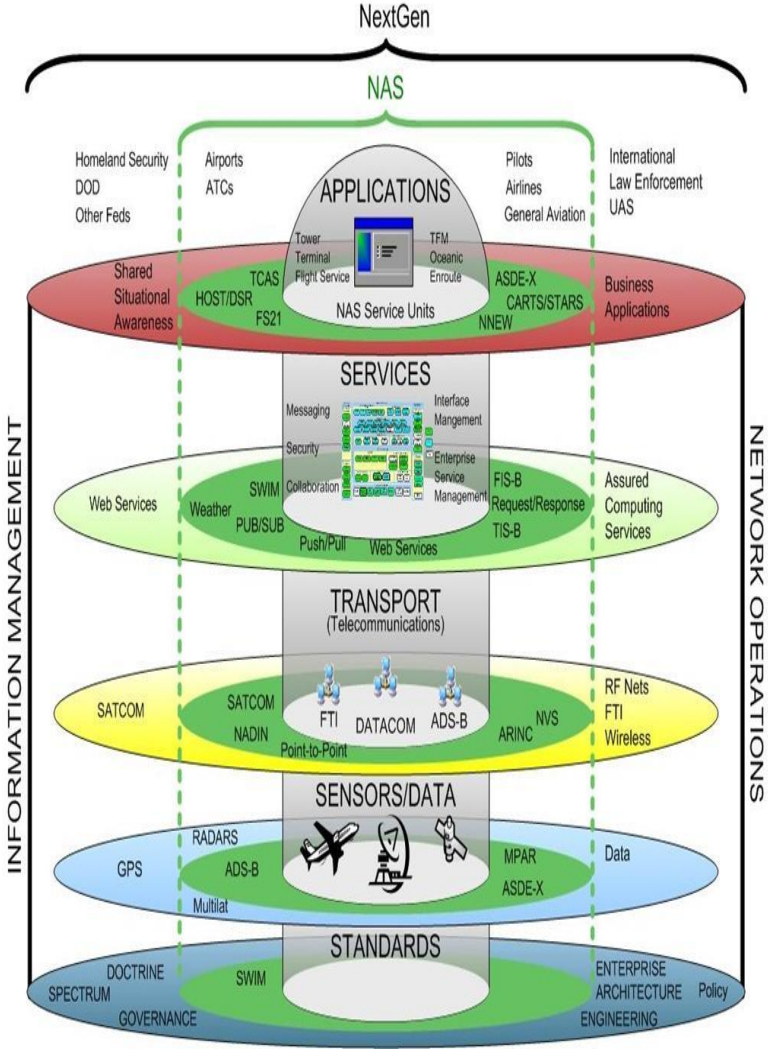


© 2014 Carnegie Mellon University

# The Challenges of Software At the Enterprise Level

**ULS systems have unprecedented scale in the following dimensions:**

- **# of lines of software code & hardware elements**
- **# of connections & interdependencies**
- **# of computational elements**
- **# of purposes & user perception of purposes**
- **# of routine processes & "emergent behaviors"**
- **# of (overlapping) policy domains & enforceable mechanisms**
- **# of people involved in some way**




**Source FAA/SEI**

**ULS systems are socio-technical ecosystems comprised of software-reliant systems, people, policies, cultures, & economics**



© 2014 Carnegie Mellon University

Federal Aviation Administration

# Why is Software V&V Becoming More Important?



© 2014 Carnegie Mellon University

# Why is Software V&V Becoming More Important?

## Threat:

Nation-state, terrorist, criminal, or rogue developer who:

- Exploits vulnerabilities remotely
- Gains control of systems through supply chain opportunities

## Vulnerabilities

- All systems, networks, and applications (Hardware & Software)
- Intentionally implanted (i.e. malicious code insertion)
- Unintentional vulnerabilities maliciously exploited (e.g., poor quality or fragile software)


© 2014 Carnegie Mellon University

# Why is Software V&V Becoming More Important?

## Importance of Software Assurance

**Software Assurance. The level of confidence that software functions as intended (and no more) and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle.**

**NDAA 2013 Section 933**

**The objective is to establish software assurance as an accepted Systems Engineering discipline within the Department of Defense.**


© 2014 Carnegie Mellon University

# Why is Software V&V Becoming More Important?

## Importance of Software Assurance


**Joint Federated Assurance Center (JFAC)**

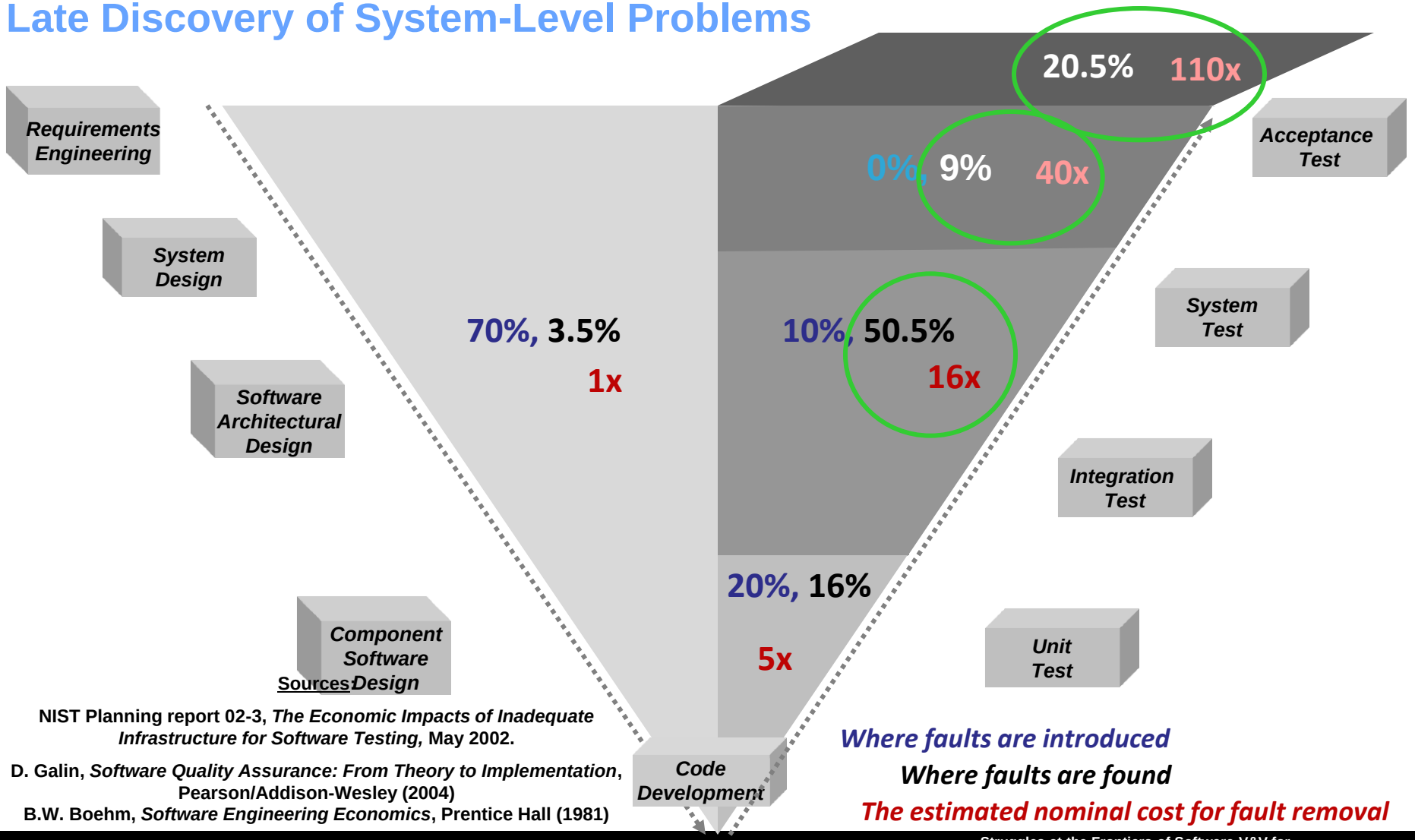
"provide for the establishment of a joint federation of capabilities to support the trusted defense system needs…to ensure security in the software and hardware developed, acquired, maintained, and used by the Department"*

*FY14 NDAA Section 937


© 2014 Carnegie Mellon University

# Why is Software V&V Becoming More Important?

## Late Discovery of System-Level Problems




Sources:

NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing,* May 2002.

D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)

B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)



© 2014 Carnegie Mellon University

# Why is Software V&V Becoming More Important?

Great capabilities, but struggling

Bifurcated community

Source: SEI



© 2014 Carnegie Mellon University

Federal Aviation Administration

# Summary

# Five Software V&V Keys for Success



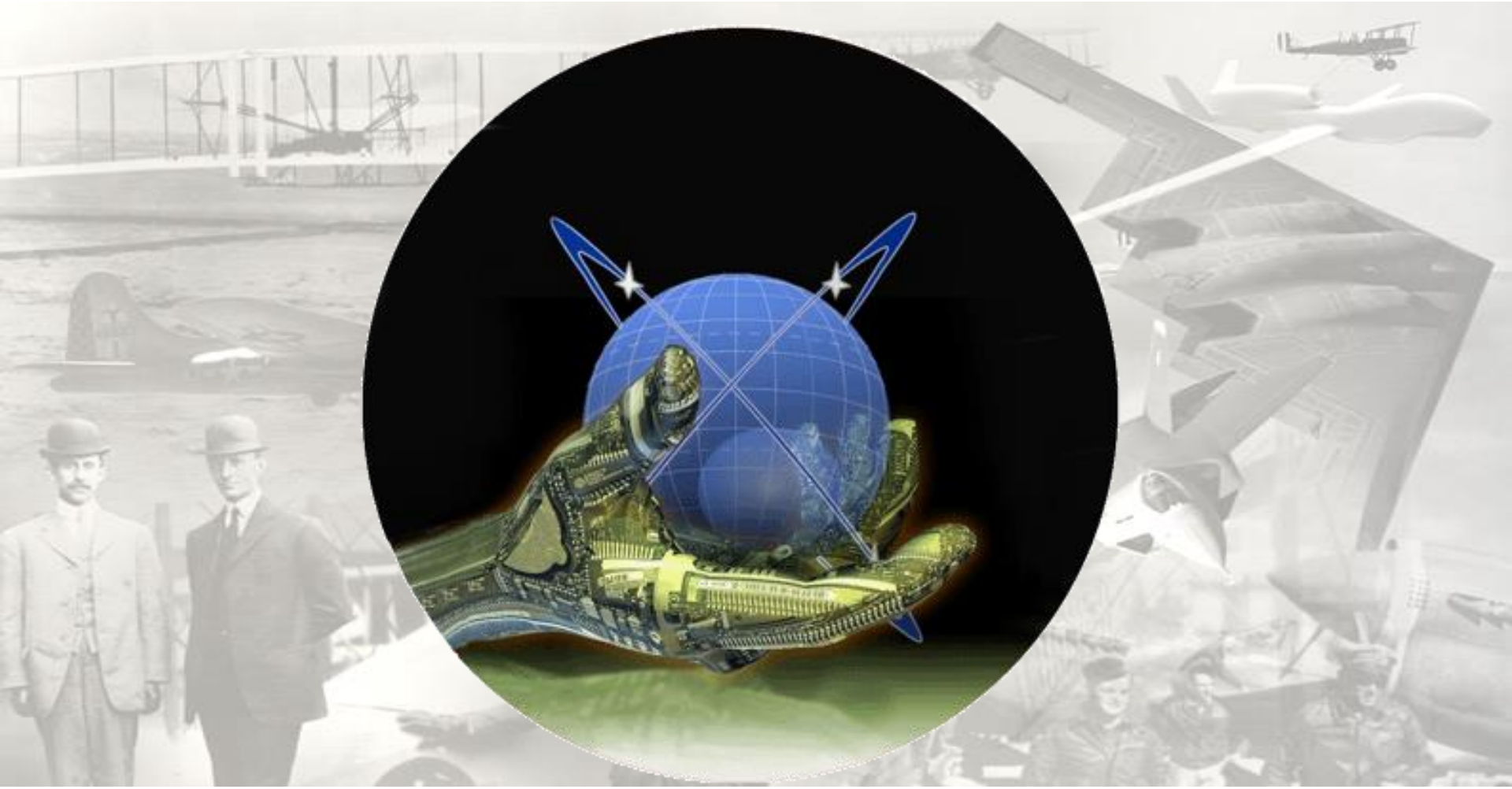
© 2014 Carnegie Mellon University

# Summary - Five Software V&V Keys for Success

1. Software V&V practices known to be effective should be consistently applied across all phases of the program life cycle.
2. Software V&V processes need to be applied to ensure requirements are well-managed throughout the life cycle, including effective translation from capabilities statements into executable requirements to achieve successful acquisition programs
3. Software V&V processes, methods and tools should be applied early in program life cycle to avoid compromising the foundation for initial requirements and architecture development.
4. Quantity and quality of V&V expertise should be sufficient to meet demands of relevant stakeholders.
5. Collaborative environments should be adequate to effectively execute V&V activities at joint capability, system of systems (SoS), and enterprise levels.


© 2014 Carnegie Mellon University

# Questions?



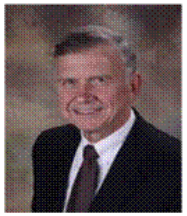


© 2014 Carnegie Mellon University

# Contact Information

Dr. Kenneth E. Nidiffer, Director of Strategic Plans

For Government Programs



Software Engineering Institute, Carnegie Mellon University

Office: + 1 703-247-1387

Fax: + 1 703-908-9235

Email: Nidiffer@sei.cmu.edu



© 2014 Carnegie Mellon University

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

Requests for permission to use or reproduce should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.



© 2014 Carnegie Mellon University